University of Glasgow | School of Computing Science

WebRTC: Media Transport and Use of RTP

Colin Perkins – University of Glasgow
Magnus Westerlund – Ericsson
Jörg Ott – Aalto University

Structure

- Core protocols
- Extensions
- Improving transport robustness
- Rate control and media adaptation
- Performance monitoring

Topologies

- RTP features and extensions suggested provide flexibility to support wide range of topologies
- Endpoints should be able to participate in any of the supported topologies, without knowing what topology is chosen
- Requirements are on browser implementation of RTP, not on middleboxes

Signalling

- Will generally ignore signalling in the following – agree what we need first, then figure out how to signal
    - SDP extensions exist to signal everything described – can probably adopt

Core Protocols

Core RTP features

- RTP is a group communication protocol, and WebRTC supports multiparty sessions

→ MUST support RTP sessions include multiple SSRCs

- Participants may have multiple media sources that need to be played out together
  - e.g., microphone and camera in a multiplexed session
  - e.g., two cameras giving different views of a scene
  - Multiple sources may come from different physical devices

→ MUST support participants that use several SSRCs simultaneously

- SSRC values need to be chosen in a scalable manner, since sessions may be large

→ MUST support random SSRC choice and SSRC collision detection/resolution

→ MAY? support signalled SSRCs via RFC 5576 (“a=ssrc:” & “a=ssrc-group:”)

- Sessions might contain RTP mixers or other middleboxes
  - Mixers have their own SSRC, and can use a CSRC list to indicate which SSRCs contributed to the mix
  - Use of CSRCs needed to detect forwarding loops caused by misconfigured RTP mixers

→ Receivers MUST support RTP packets containing CSRC lists, and MUST handle RTCP packets relating to CSRCs

RTCP

- RTCP is an essential component of RTP

→ MUST implement RTCP

- Standard RTCP packet types are defined in RFC 3550: SR, RR, SDES, APP, and BYE

→ MUST implement SR, RR, SDES, BYE
→ Other RTCP packets OPTIONAL, but MUST ignore unsupported packet types

- Lip-sync requires correct SR & CNAME packets

→ MUST support lip-synchronisation

- RTCP timing rules scale transmission interval according to size of the session; randomisation and timer reconsideration avoid traffic bursts

→ MUST scale RTCP interval according to number of SSRCs in RTP session

→ MUST randomise RTCP interval and MUST implement timer reconsideration

- RTCP bandwidth needs to be configured based on codec choice, and desire for rapid feedback

→ MUST support configurable RTCP bandwidth

Choice of RTP Profile (1)

- Several RTP profiles defined:
  - RTP/AVP
  - RTP/AVPF
  - RTP/SAVP
  - RTP/SAVPF
- RTP/AVP is the baseline

Choice of RTP Profile (2)

- Several RTP profiles defined:
  - RTP/AVP
  - RTP/AVPF
  - RTP/SAVP
  - RTP/SAVPF

- Rapid feedback profile
- Improved RTCP timer model
- More flexible; allows transmission in response to events, not just the regular periodic schedule
- Compatible with RTP/AVP
  - Designed so changes to timing model kept same average behaviour, and so don't disrupt RTP/AVP endpoints
  - Requires parameters, e.g., trr-int, to be set appropriately

→ Do we need to give guidance on parameter choice?

- Required to use conferencing or codec control extensions
  - FIR, PLI, SLI, RPSI, TMMBR, CCM, etc.

Choice of RTP Profile (3)

- Several RTP profiles defined:
  - RTP/AVP
  - RTP/AVPF
  - RTP/SAVP
  - RTP/SAVPF

- Secure RTP and RTCP
- Media encryption, integrity and replay protection

→ Do we need to give guidance on crypo transforms, etc? (not how to negotiate, but what values to negotiate)

Choice of RTP Profile (4)

- Several RTP profiles defined:
  - RTP/AVP
  - RTP/AVPF
  - RTP/SAVP
  - RTP/SAVPF

- The combination of RTP/AVPF and RTP/SAVP profiles
- Rapid feedback extensions
- Security

→ MUST use RTP/SAVPF profile

Discussion of keying mechanisms needed, but not in this draft

Choice of RTP Payload Formats (1)

- Endpoints can signal support for multiple payload types
  - Different codecs, or different configurations of same codec; each MUST use a different RTP payload type number
- When are encoders allowed to change encoding? Should receivers accept payload type changes at arbitrary times?

→ REQUIRED to accept data in any signalled payload type at any time, unless previously signalled limitations on decoding capability

If audio and video are multiplexed in one RTP session, an SSRC can only switch between payload types of the same media type

Signalling before payload type change not required, to allow codec changes to be used for congestion control

Choice of RTP Payload Formats (2)

- If this group mandates particular codecs, this draft will reference the draft making the recommendation

→ This draft will not mandate codecs

- As additional input to the codec discussion, RFC 3551, section 6 says:

“Audio applications operating under this profile SHOULD, at a minimum, be able to send and/or receive payload types 0 (PCMU) and 5 (DVI4).”

This is implicitly adopted by our choice of the RTP/SAVPF profile

RTP Session Multiplexing

- Participants that see a common shared SSRC space form an RTP session; RTP sessions can span multiple transport connections
- Legacy implementations follow RFC 3551, and send audio and video on separate RTP sessions
  - i.e., using separate UDP ports

RFC 3551 states: "Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session, but multiple RTP sessions MAY be used in parallel to send multiple media types."

→ REQUIRED to support separate media on separate RTP session, for compatibility

- Multiple transport layer flows problematic for NAT traversal, so desirable to reduce number of transport flows
  - Can use a single RTP session, with multiple media types (violates "SHALL NOT" in RFC 3551, leads to anomalous RTCP behaviour, and has other limitations, but workable in the current WebRTC use cases)
  - Desirable to keep the RTP session distinction, while reducing the number of transport ports, to support widest range of RTP features, and for ease of gateway operation

→ REQUIRED to support multiplexing a multimedia session onto a single RTP session (draft-… ?)

→ RECOMMENDED to support multiplexing multiple RTP sessions onto a single transport flow (e.g., draft-westerlund-avtcore-transport-multiplexing-02)

RTP and RTCP Multiplexing

- RTP and RTCP historically run on separate ports, but this can be problematic for NAT traversal
- RFC 5761 describes how to multiplex RTP and RTCP on a single port
  - Reduces number of NAT bindings, provides keep-alive traffic on the port
  - Restricts usable RTP payload type numbers to avoid collisions with RTCP
  - Requires SDP signalling

→ REQUIRED to implement RFC 5761

→ Should WebRTC implementations be required to support use of RTP and RTCP on separate ports, for legacy interoperability?

Symmetric RTP/RTCP

- RFC 4961 defines symmetric RTP
- This requires the same port be used to transmit and receive RTP packets
  - RTCP also required to use the same port when sending and receiving (RFC4961 assumes, but does not require, this to differ from the RTP port)
- Simplifies NAT traversal

→ REQUIRED to implement RFC 4961

Reduced Size RTCP

- RTCP is sent as compound packets, starting with an SR or RR packet
  - Needed in regular RTCP reports for monitoring reception quality; lip-sync
  - Unnecessary and wastes bandwidth in rapid feedback requests
- RFC 5506 discusses use of non-compound RTCP
  - Can only be used with RTP/AVPF and RTP/SAVPF, and MUST NOT be used for regular reports
  - Requires SDP signalling (“a=rtcp-rsize”)

→ REQUIRED to implement RFC 5506

→ Potentially non-compatible with legacy devices – should we require ability to send rapid feedback in compound packets?

Generation of the RTCP CNAME

- RTCP CNAME provides a consistent identifier for an endpoint during a session
  - SSRC values can change due to collisions, CNAME remains constant
  - Used to associate flows for synchronisation
- RFC 3550 CNAME format is “doe@192.0.2.89” – privacy concerns; problematic with NAT
- RFC 6222 defines a short-term persistent CNAME based on hash of time-of-day and MAC address
  - RFC6222, section 4.2, method (b)

→ REQUIRED to implement RFC 6222

  - The RFC3550 format is “SHOULD”, legacy compatibility concerns limited

Extensions

Conferencing Extensions

- Multiparty conferences often implemented using centralised servers
- These will work using only standard RFC 3550 features, but will perform poorly
- Rapid feedback and Codec Control Messages greatly improve performance

→ REQUIRED to implement RFC 5104, but which options?

Temporary Maximum Media Stream Bit Rate Request

- A receiver or middlebox may be aware of rate limit on available capacity
- Desirable to restrict sending rate to this limit
    - e.g., a centralised conferencing middlebox might estimate the available bandwidth to all receivers, and enforce a limit on sending rate to match slowest receiver
    - Temporary Maximum Media Stream Bit Rate (TMMBR) Request enables this limit [RFC 5104 sections 3.5.4 and 4.2.1]

→ Senders are REQUIRED to respect TMMBR requests

→ Receivers MAY send TMMBR requests

Full Intra Request (FIR)

- Centralised multiparty calls often implemented by switching between sources
  - Naïve implementation sends video that cannot be immediately decoded to receivers, since switch into stream doesn’t occur at decoder refresh point
  - Greatly improves user experience if RTP mixer or switching MCU sends a full intra request to the sender when switching streams, to trigger refresh [Codec control messages; RFC 5104 sections 3.5.1 and 4.3.1]
  - Essential to make centralised switching viable for multiparty conferences

→ Senders are REQUIRED to support full intra requests

Semantic Loss Tolerance (1)

- Packet loss can cause receivers to lose decoder context, and be unable to render video
  - Potentially long-lived situation, depending on frequency of insertion of decoder refresh points by sender
  - Picture Loss Indication message signals this to the sender; semantically different to FIR due to implication that network problems occurring – but similar response from sender
  - Not essential, but can improve user experience on lossy networks

→ Senders and receivers RECOMMENDED to support picture loss indication messages [RFC 4585, secton 6.3.1]

Semantic Loss Tolerance (2)

- More limited picture disruption can also occur
  - Slice loss indication message [RFC 4585 section 6.3.2] tells encoder that one or more consecutive macroblock(s) in scan order have been lost
  - Encoder can then repair effects using a partial intra refresh, for example
  - Can offer limited improvement to user experience on lossy networks

→ Support for slice loss indication messages is OPTIONAL

  - Reference picture selection [RFC 4585 section 6.3.3] allow encoders to request an alternate reference picture be used for future coding, to hide effects of packet loss

→ Support for reference picture selection is OPTIONAL

  - These are useful loss-tolerance mechanisms, but not essential

Controlling Codec Operation

- How to control codec features such as frame rate or video resolution?
  - draft-westerlund-rtcweb-codec-control-00 – signals envelope within which codecs must operate in JSEP; codec operation point RTCP extension for ongoing adaptation
  - draft-alvestrand-rtcweb-resolution-00 – uses SDP for initial and in-call negotiation of operating point; adopts the codec operation point RTCP extension as an OPTIONAL feature

→ Need to decide if this feature is needed, which mechanism to use?

Other Codec Control Messages

- Other standardised codec control messages:
  - H.271 video back channel message
  - Temporal-spatial trade-off request
  - (Both defined in RFC 5104)

→ Not useful in WebRTC context?

Header Extensions

- RFC 3550 allows for RTP header extensions, but with poorly defined semantics
- RFC 5285 extends this definition, to allow stackable, clearly defined, header extensions
  - Requires SDP signalling (“a=extmap:”) to map from short extension names in RTP packets to extension definitions
  - Describes how to use header extensions, but doesn’t define any extensions

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|1|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0xBE     |      0xDE     |    header extension length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       header extension                        |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          payload data                         |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

→ Header extensions OPTIONAL, but if supported MUST follow RFC 5285; unknown extensions MUST be ignored

- For compatibility with RFC 3550 and legacy implementations, RFC 5285 states:

“header extensions MUST only be used for data that can safely be ignored by the recipient without affecting interoperability, and MUST NOT be used when the presence of the extension has changed the form or nature of the rest of the packet in a way that is not compatible with the way the stream is signalled”

Rapid Synchronisation

- Synchronisation between flows enabled by information in RTCP SR and SDES CNAME packets
- Can be slow to acquire sync, if first RTCP packet is lost, or in groups where initial RTCP delay is large
- RFC 6051 conveys synchronisation information in header extension, for rapid synchronisation
- Also defines a feedback message, RTCP-SR-REQ, to solicit an RTCP SR packet, if synchronisation lost

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    RC   |   PT=SR=200   |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC of sender                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|              NTP timestamp, most significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             NTP timestamp, least significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RTP timestamp                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     sender's packet count                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      sender's octet count                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_1 (SSRC of first source)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| fraction lost |       cumulative number of packets lost       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           extended highest sequence number received           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      interarrival jitter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         last SR (LSR)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   delay since last SR (DLSR)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

→ RECOMMENDED to implement RFC 6051

Client-to-Mixer Audio Level

- When using an RTP mixer, sources to be mixed are often determined by audio activity
- RFC 6464 defines RTP header extension to convey audio level information from client to mixer
- Allows mixer to determine active streams, without having to decode all streams
    - Can also reduce receiver decoding requirements in full mesh conferences

→ Senders RECOMMENDED to implement RFC 6464; RTP header extension MUST be encrypted using draft-ietf-avtcore-srtp-encrypted-header-ext-01 if used

Mixer-to-Client Audio Level

- CSRC list indicates SSRCs that have contributed to a mixed stream
- RFC 6465 defines RTP header extension to convey audio levels for components of the mix
- Can be used to provide UI cues for audio levels

→ RFC 6464 support is OPTIONAL for receivers; header extension MUST be encrypted using draft-ietf-avtcore-srtp-encrypted-header-ext-01 if used

Other RTP Header Extensions

- Two other standardised RTP header extensions:
  - RFC 5450 – transmission time offset for improved jitter calculation
  - RFC 5484 – SMPTE times codes

→ Not useful in WebRTC context

Improving Transport Robustness

Improving Transport Robustness

- Retransmission
- Forward Error Correction (FEC)
  - Basic redundancy
  - Block-based FEC

- All potentially add packets during congestion
  - Use to repair loss must be balanced against potential to cause loss

→ Use of retransmission or FEC MUST be coupled with congestion awareness

Retransmission

- RTP/AVPF profile incorporates NACK feedback
- RFC 4588 provides mechanisms to retransmit lost RTP packets, based on this feedback
    - Independent of payload type
    - 1 RTT to repair loss, so only suitable if RTT low relative to playout buffer, and sufficient delay budget
    - Can gives valuable improvement to user experience in many scenarios

→ RECOMMENDED to implement RFC 4588 and NACK feedback

Basic Redundancy

- Wide variety of basic redundancy mechanisms
  - Redundant encoding within RTP payload format (e.g., in AMR-WB and G.719)

→ MAY be used if supported by payload format

  - RTP payload format for redundant audio data (RFC 2198)

→ RECOMMENDED for use with text conferencing; payload-format specific mechanisms better for audio

  - Stream duplication
  - Proactive transmission of redundant packets

→ Not appropriate for WebRTC

Block-based FEC

- Several alternatives
  - RTP Payload Format for Generic FEC
    - RFC 2733 – basic parity FEC; breaks if header extensions or CSRC lists are used (variants are also widely used – SMPTE 2022-1, Pro-MPEG CoP)
    - RFC 5109 – fixes problems with RFC 2733, adds unequal error protection
  - RTP Payload Format for 1-D Interleaved Parity FEC
    - RFC 6015 – fixed version of RFC 2733, without unequal error protection
  - FEC Framework
    - draft-ietf-fecframe-framework-…
    - Which FEC schemes to use (many have known IPR)?

→ Interworking with legacy problematic, due to wide variation in practice; RFC 6015 seems best trade-off if we are to recommend anything

Rate Control and Media Adaptation

Congestion Control

- Clear that congestion control is needed, premature to mandate an algorithm
- But, some limit needed to avoid congestion collapse

→ Implementations will be REQUIRED to implement draft-perkins-avtcore-rtp-circuit-breakers (assuming completed in AVTCORE)

→ Implementations are REQUIRED to support signalled hard limits on bandwidth use

→ Should we require support for (trying to use) ECN for RTP flows? A potential driver for ECN deployment, but introduces complexity

Performance Monitoring

Performance Monitoring

- Basic RTCP reports on RTT, packet loss, and jitter
- RTCP Extended Reports (XR) provide *many* more metrics

→ Should we require support for any RTCP XR metrics? (no requirements in use cases draft as yet, but draft-huang-rtcweb-monitoring-00 has some suggestions)

Further Discussion?